ministère des PTT

92172

SUP TÉLÉCOM

ENST - 87C004

# Using CS-PN Software for Protocol Specification, Validation and Evaluation

Alexandre ZENIE, Jean-Pierre LUGUERN
Département Informatique

1987

ECOLE NATIONALE SUPERIEURE DES TELECOMMUNICATIONS

ENST - 87C004

# Using CS-PN Software for Protocol Specification, Validation and Evaluation

Alexandre ZENIE, Jean-Pierre LUGUERN
Département INFORMATIQUE

1987

# USING CS-PN SOFTWARE FOR PROTOCOL SPECIFICATION, VALIDATION AND EVALUATION

I. Alexandre ZENIE

J. Pierre LUGUERN

1987

- ENST - MASI - BULL -

## RESUME

Il est important d'analyser le parallélisme entre les différentes actions d'un protocole d'une part, et de valider le fonctionnement et d'évaluer les performances de protocoles d'autre part.

Le logiciel CS-PN, dans sa version actuelle [20], permet de spécifier les protocoles utilisés en systèmes répartis, de valider le comportement qualitatif et d'évaluer ensuite les performances des protocoles de façon modulaire et analytique en s'appuyant sur l'outil formel des réseaux de Petri stochastiques colorés [19].

Nous décrivons d'une part la spécification au moyen du langage C-DEOL [7], et d'autre part la vérification, la validation et ensuite l'évaluation qui constituent les différentes étapes du logiciel CS-PN [20].

Nous appliquons ce logiciel au protocole Blesse/Attend [12], développé à l'ENST, qui gère la prévention dynamique d'interblocage entre transactions concurrentes dans un système de gestion de bases de données réparties (Système R*).

Cet article fait parti des actes du congrès "4th International Workshop on Software, Specification and Design" sponsorisé par IEEE Computer Society, ACM SIGSOFT, LCRST (Japon), Alvey Directorate (Grande Bretagne), par l'Agence de l'Informatique et par l'AFCET (France).

Ce congrès a eu lieu les 3-4 Avril 1987 à Monterey en Californie.

# USING CS-PN SOFTWARE FOR PROTOCOL SPECIFICATION, VALIDATION AND EVALUATION

I. Alexandre ZENIE & *

J. Pierre LUGUERN & #

& Ecole Nationale Supérieure des Télécommunications, Département Informatique, 46, rue Barrault, 75634 PARIS Cedex 13, FRANCE
* Université Pierre et Marie Curie, Laboratoire MASI (CNRS/UA 818), 4, place Jussieu, 75230 PARIS Cedex 5, FRANCE
# BULL, Département DEDL/ DIT, 94, avenue Gambetta, 75020 PARIS, FRANCE

## RESUME

It is important to analyze the paralellism between a protocol's different actions, on the one hand, and to validate protocol function and evaluate performances, on the other.
The CS-PN software enables one to specify the protocols used in distributed systems, to validate qualitative behavior, and then, by relying on the CS-PN formal tool, to validate protocol performances by modular and analytic means [19].

We intend to describe the specification, verification, validation, and afterwards, evaluation, which make up the different steps of the CS-PN software.
We will apply this software to a Wound/ Wait protocol [12] which manages the dynamic prevention between concurrent transactions in a distributed data management system (system R*).

## I - INTRODUCTION

The SPN tool [6,14,15] permits the modeling, analysis and evaluation of computer systems' performances [16].
Several software rely on this tool, we cite in particular GSPN (TORINO University) [1,2], RDPS (CERCI-CNAM-CIMSA) [5], ESPN (DUKE University) [4], and SAN (MICHIGAN University) [13].

The colored stochastic Petri nets tool is a natural extension of the preceding tool, adapted to distributed systems and protocols, because the color conveniently takes into account the numerous sites, transactions, granules and messages [18].
By relying on this tool, and thanks to the collaboration of the CIMSA-SINTRA (Velizy,FRANCE), MASI (P.M. CURIE University, FRANCE) and AARHUS University (DENMARK) laboratories, the CS-PN software developed at ENST allows the specification (with the aid of the C-DEOL language) [7], verification (by a syntactic analysis of invariants) [3,8,11], validation (by a semantic analysis of the colored states graph) [9,10], and performance evaluation (by a quantitative analysis of the colored Markovian chain isomorph to the graph of colored states) [17] of protocols used in the distributed systems.

The CS-PN software is applied to a Wound/ Wait protocol decomposable into two principal modules: request or couple (transaction, granule) treatment module and wound treatment module.
Each module is specified, verified, validated, and then evaluated separately, to eventually deduce a verification, validation and evaluation of the complete protocol.

## II - THE CS-PN SOFTWARE

The CS-PN software is developed on a VAX 780-11 machine under VMS4.3 in Pascal language.
This consists in specifying a protocol with the aid of a C-DEOL language (a Colored Dependability Evaluation Oriented Language) in a predefined file named *<file_name>*•DEO, by using a classic VAX-VMS editor (EDIT,EVE). The file *<file_name>*•DEO is compiled in order to verify, validate and then evaluate.

### 1. SPECIFICATION

While relying on the colored stochastic Petri nets, the specification in C-DEOL uses an editor to create a specification file to be compiled.

#### 1.1° The C-DEOL Language

Specification in C-DEOL breaks down into four parts:
-The declaration of different objects intervening in the specification. These are primarily the places and transitions and, with the introduction of colors, the types or the color sets associated with places, transitions, variables, and functions.
-Topological specification, in other words, the static specification of the preconditions and postconditions relative to each of the transitions.
-The initial dynamic specification , in other words, the specification of the initial marking of each of the places concerned.
-The timed specification, which is to say, the specification of the firing rates of each transition. The firing rates are associated with the firing instances of transitions, which are exponentially distributed aleatory functions.

The general structure of a specification in C-DEOL is:

```
STUDY study_name ;
/*Declarations*/
TYPE type_name = [color 1, ... ,color c] ;
PLACE place_name 1, ... , place_name n [: type_name] ;
TRANS trans_name 1, ... , trans_name m [: type_name] ;
VAR variable_name 1, ... , variable_name v : type_name ;
FUNCTION function_name (type_name 1) ---> type_name 2 ;
f_instruction 1 ;
...
f_instruction k ;
/* An f_instruction expresses itself in the following manner:
color_name 1 ---> color_name 2, ... , color_name c ;
where color_name 1 must belong to type_name 1, and color_name 2, ... ,
      color_name c must belong to type_name 2 */
EF ;
/*Static Specification*/
IF precondition THEN postcondition ;
/*A precondition expresses itself in the following manner:
term 1 AND term 2 AND ... AND term n
where term is of the form:
place_name [ (color) ] [valuation] */
/*A postcondition expresses itself in the following manner:
trans_name [ (color) ] : term 1 AND term 2 AND ... AND term n */
/*Specification of the Initial Marking*/

INITIAL
place_name [ (color) ] := mark_number ;
END
/*Specification of Transition Firing Rates*/
RATES
trans_name [ (color) ] : λ . [ α . m (place_name1)[(color)] + β . m (place_name2) [ (color) ] ] ;
END
END
```

The type declaration **TYPE**, permits the definition of a color set. This set will be associated with one or several places, transitions, and variables.
*type_name* and *color* are the identifiers. The *color* identifier is implicitly defined as being one color.
Declaration **PLACE** (respectively **TRANS**) allows the definition of one or several places (respectively transitions).
*place_name* (respectively *trans_name*) and *type_name* are the identifiers. The *name_type* identifier must obligatorily have been defined in a type declaration. The mention *type_name* is optional. It serves to associate a set of colors, if one exists, with one or several places (respectively transitions).
The **VAR** declaration, allows the definition of one or several variables by specifying their colors set.
*variable_name* and *type_name* are identifiers. The *type_name* identifier, must obligatorily have been defined in a type declaration.
The declaration of a function **FUNCTION**, enables one to define a correspondence between each of the colors of two sets. A color from the starting set can have several colors of the arrival set for an image.
In static specification, *term*, represents a part of the precondition respectively of the postcondition: a place-marking condition.
Two optional arguments are possible for specifying this condition:

- *color* enables one to specify the place's color marks used by the precondition respectively postcondition.
  This optional argument also serves to specify the color of a fireable transition. The possible color argument forms are:
  . EMPTY
  . *color_name*
  . *variable_name*
  . EVERY *variable_name*
  . *function_name* (*color_name*)
  . *function_name* (*variable_name*)

- *valuation* permits one to assign a coefficient to the arc etiquette inputting the transition respectively outputting the transition. It can have three forms:
  - = k : test if the marking of the place in input is superior or equal to k. during transition firing, the marks will be removed from the marking.
  = 0 : test if the marking of the place in input is null.
  + = k : increments the marking of the place in output by k marks.
  By default the valuation is equal to - = 1 respectively + = 1.

In the specification of the initial marking of a place: *place_name* and *color* are the identifiers, *mark_number* is an unsigned natural number.
The *color* optional argument allows one to specify the color of the initial marks.
In the specification of the transition firing rates: *trans_name*, *place_name* and *color* are identifiers, λ, α, β are the reals, and **m** designates the place marking.
The transition firing-rate depends on the optional *color* argument, and expresses itself either by means of a real constant, or with the aid of a linear function of the marking of one or several places.
If the real constant is be infinite, the corresponding transition is then immediate.

1.2° Editor

Any editor from the VAX-VMS, EDIT or EVE, for example, allows the specification of one or

several modules in a file by means of the C-DEOL language, predefined <*file_name*>•DEO with the aid of command language DCL.The modules can have one or several places in common, which makes the specification modular.
A module must begin with the key word **STUDY** and terminate with the key word **END**. The key words **TYPE, VAR, FUNCTION, INITIAL,** and **RATES** are optional. Commentaries are placed between two particular symbols: " /* " for the beginning of the commentary, and " */ " for the end of the commentary.

1.3° Compiler

The different modules can be compiled either separately or with a link. The three principle compiler functions are:
The recognition of the C-DEOL language, error handling, and the generation of a file codification <*file_name*>•<*study_name*>•DEO:

-Codification of declarations ( <*file_name*>•<*study_name*>•IMP for the edition of the identifier table, <*file_name*>•<*study_name*>•IDE for the table of identifications, <*file_name*>•<*study_name*>•DES for the study description ).
-Codification of the static specification ( <*file_name*>•<*study_name*>•NET ).
-Codification of the initial marking specification ( <*file_name*>•<*study_name*>•INI ).
-Codification of the transition firing-rate specification ( <*file_name*>•<*study_name*>•RAT ).

If the compiler detects the slightest error it generates the file <*file_name*>•<*study_name*>•IER containing the generic of each error.
If the detected error is not serious (warning of level 1 and 2) all the files are created, otherwise only the " •IER " file will be created.
The compiler can break itself down into four packages:

- Lexical analysis
-Syntactic analysis (parser) and the generation of information structures in internal format.
- identifier handling.
- error handling.

The lexical analyzer acts as an interface between the file source " •DEO " and the syntactical analyzer: It reads the file source character by character, and recognizes the lexical units (tokens) in order to send the syntactical analyzer:

. The chain of characters containing the lexical unit. It is a global variable, implanted in the syntactical analyzer and,
. The lexical unit code.

As soon as the syntactical analyzer ends the treatment of a lexical unit, it calls the lexical analyzer to obtain another lexical unit. The call to the lexical analyzer is thus cyclic.
When the syntactical analyzer meets an identifier (a particular lexical unit), it calls the identifier handler, which entrusts itself with either filing this identifier or searching for it.
Filing takes place by attributing an identifying number, and an address in the identifier tables.
The search for an identifier takes place through a test of the table of identifiers.
When the syntactical analyzer detects an error, it calls the error handler, which takes over editing the corresponding error message.
The syntactical analyzer is thus the kernel of the compiler, it verifies that the C-DEOL language's syntax is well obeyed and then generates the different files containing the codification of the specification.

## 2. VERIFICATION

The principle of verification is based on the resolution of a linear equation of the form $v^t \cdot W = 0^t$, for linear invariants v at the places, and of a linear equation of the form $W \cdot w = 0$ for the linear invariants w on the transitions. W being the incidence matrix built from postconditions and preconditions.
The v (respectively w) obtained syntactic invariants with minimal supports, express the conservation relations of marks in the places (respectively of repetition of transition firing) and thus enable one to verify that specification realizes the properties required of the protocol (the property of mutual exclusion, the absence of certain deadlocks, the repetition of certain actions without modifying the rest of the protocol).
If verification does not achieve the hoped-for results, it is then possible to start the specification once again.

## 3. VALIDATION

The dynamics of specified protocol are characterized by the accessible markings graph, from the specified initial marking.
The construction of this graph consists in determining whether an obtained marking is truly accessible from the initial marking, and if it is equivalent to an already extant marking.
The applied equivalence relation allows one to regroup several markings in a single c-marking (color marking). Thus, a color set is associated with each c-marking of the graph of accessible markings, and another color set, is associated with each arc linking two c-markings and characterizing the firing of a transition.
The principle of the equivalency relation is based on the permutation of the different colors belonging to each color set.
The advantage of such a construction is the optimization of the number of c-markings and the structuring of the different states of the specified protocol.
This validation principle allows one to deduce the boundedness properties (which express the fact that a limited number of sites, transactions, granules, and messages are enough for

specification) of vivacity (which signifies the absence of deadlock) and of reinitializability (which validates the possibility of restarting the specific protocol from the initial marking, and the necessity of reinitializing the protocol for any repetition) for each module of the " •DEO " file and for the complete protocol in particular.
In any case, the protocol's validation can be deduced from the validations of a protocol's different modules, hence the advantage of this principle's modularity.
This method of validation allows one to validate the good functioning of a protocol's different modules, the functioning of the protocol itself, and the operation of the service which each module and protocol must render.
If validation does not reach the protocol's wished-for properties, then one or several specification modules are restarted.

## 4. PERFORMANCE EVALUATION

Once the protocol is validated, it becomes possible to evaluate the performance of each of the specification's modules.
The evaluation principle is based on the isomorphism between the accessible c-markings graph, constructed during the previous step, and the Markov chain, obtained by replacing the names of transitions linking the c-markings with rates of associated firings.
The evaluation thus consists in building the matrix A of transitions expressing the probability of the passage of one c-marking to another, and resolving a linear equation of the form $\mathbb{P}^{*t} \cdot A = 0^t$ such that the vector $\mathbb{P}^*$ norm is equal to 1. Vector $\mathbb{P}^*$ designates the probabilities of the different c-markings in a steady state.
To resolve such an equation, one uses the same Farkas algorithm used to find the invariants in verification.
The c-markings probability vector, thus obtained, enables one to deduce the performance criteria associated with each module and protocol, such as the mean number of marks in a place, the mean firing frequency of a transition, and the mean mark sojourn time in a place, relative to a color or for any color.
Several file codifications *<file_name>•<study_name>*•RAT are generated with the aid of separated compilation, and evaluation is applied to each codification, in order to deduce the performance criteria associated with several specifications of the transition firing-rate of the same static specification.

## III - APPLICATION TO DS

The distributed data base is made up of a set of granules distributed uniquely between geographically situated network sites.
Each granule is unique and local to a site (system R*). The conflictual access to one granule by several sites requires a serializable coherence control protocol.

## 1. THE WOUND/ WAIT PROTOCOL

This protocol's principal is based on the timestamp of transactions which allow conflicts to be managed from the only local state of the granule in question.
It is also based on two-phase locking, where each transaction locks the granule in the compatible mode, before carrying out an operation, and unlocks it, after the end of the operation by not accepting any new locking requests following the first unlocking.
The Wound/ Wait protocol introduces a transaction's wound and an FWAIT queue with priority to the oldest transactions.
If there is conflict between a Tr transaction, which asks to lock granule g while it is already locked in an incompatible mode by transaction Tl, the Wound/ Wait protocol will use the following principle:

**IF** Tr is older than Tl **THEN** Tl is wounded **ELSE** Tr waits;

as well as the two following rules:

i) a healthy transaction can only be placed on queue from the end or abort of another transaction, one either older than it, or a wounded transaction.

ii) a wounded transaction, is only authorized to await the unlocking of a granule if it is older than the lock transactions, and if no transaction older than it asks for g to be unlocked in an incompatible mode.

The Wound/ Wait protocol can be broken down into two principal modules:

### 1.1° The Request Treatment Module

Either a transaction Tr asking to lock granule g in mode m (shared or exclusive).
* If g is free, there is no conflict and Tr is authorized to lock g.
* If g is locked:
  • Either Tr is older than all the FWAIT transactions or FWAIT is empty, Tr is then allowed to try to lock g for itself.
    - If the present lock is compatible with Tr's request, Tr is authorized to lock g for itself.
    - Otherwise there is conflict. Let Tl be the set of transactions having locked the granule:
      • If no transaction Tl is older than Tr, Tr is placed on queue at the ending or roll back of Tl transactions. All Tl transactions are then wounded (rule i).
      • Otherwise:
        . If Tr is wounded, it is rolled back (rule ii).
        . If Tr is not wounded, it is authorized to wait. All Tl transactions younger than Tr are then wounded (rule i).
  • Otherwise Tr is never allowed to try and wound transactions having locked g.
    - If Tr is not wounded it is authorized to queue.
    - Otherwise:

- If Tr's wait respects rule ii, it is authorized to queue.
- Otherwise Tr is rolled back.

In every case, if Tr has been placed in queue or authorized to lock g, the wound transactions in queue, which no longer respect rule ii, must be rolled back.

## 1.2° The Wound Treatment Module

The Wound/ Wait protocol's delicate point, is situated at the level of treatment for a transaction wound.
When a transaction Ta wounds transaction Tv, it only wounds one of its agents. However, this agent can no longer be active on this site. The wound has conflict handling for a goal; knowing whether the transaction must be rolled back or not (rule ii). It is only of interest to the active agents and must thus send them the message. When a site must send a wound, the following principle is applied to each granule which the transaction has already locked:

- If the local agent is terminated, the message is transmitted to the invoking agent.
- If the local agent awaits the termination of an agent, the message is sent to the last agent which it initialized.

'Optimization' is sending a message from a site. If locally, the transaction has not been wounded. In any case, this does not prevent several emissions for the same transaction. This is due simply to transmission delays and to site desynchronization.
When a site receives a wound message:

- If the local agent is already wounded locally, there is no treatment to undertake.
- Otherwise the local agent passes into the wounded state.
    - If the local agent is in unlock queue of another granule:
        - If it does not respect rule ii, it is rolled back.
        - Otherwise its queue pursual is authorized.

## 2. SPECIFICATION

The specification of the Wound/ Wait protocol breaks down into four modules:

### 2.1° Request Treatment Module

**STUDY** Request_Treatment_Module ;
**TYPE** TRANSACTIONS = [ t ]; GRANULES = [ g ]; SITES = [ s ]; REQUESTS = [ tg ]; AGENTS = [ ts ]; TRANSCONFLICT = [ tgt']; EVENTS = [t die,t term];
**PLACE** GT : TRANSACTIONS; RG : GRANULES; TR, LOCK, CF, NEXT, FWAIT, TWAIT, TLOCK : REQUESTS; WOUND, TW : AGENTS; WK : TRANSCONFLICT; UNLOCK : EVENTS;
**TRANS** $\alpha_1$ : TRANSACTIONS; $\alpha_3$, $\alpha_5$, $\alpha_8$, $\alpha_9$ : REQUESTS; $\alpha_2$, $\alpha_4$, $\alpha_6$, $\alpha_7$, $\alpha_{10}$: TRANSCONFLICT;
**FUNCTION** INIT ( TRANSACTIONS ) ---> REQUESTS;
t ---> tg;
**EF;**
**FUNCTION** S ( GRANULES ) ---> SITES;
g ---> s;
**EF;**
**FUNCTION** V ( TRANSACTIONS ) ---> TRANSACTIONS;
$t'_{|t'< \text{EVERY } t}$ ---> t';
**EF;**
/* EVERY$_1$ designates the projection onto the first component of the function EVERY, EVERY$_{1|c}$ designates the projection onto the first component, verifying condition c of function EVERY */
**IF** GT(t) **THEN** $\alpha_1$ : TR(INIT(t));
**IF** RG(g) **AND** TR(tg) **AND** LOCK(EVERY$_1$ t'g) **THEN** $\alpha_2$ : CF(tg) **AND** LOCK(t'g);
**IF** RG(g) **AND** TR(tg) **AND** LOCK(EVERY$_1$ t'g)=0 **THEN** $\alpha_3$ : RG(g) **AND** NEXT(tg) **AND** LOCK(tg);
**IF** FWAIT(V(t')g) **AND** CF(tg) **AND** FWAIT(EVERY$_{1|t''<t'}$ t''g)=0 **THEN** $\alpha_4$(t'<t) : FWAIT(t'g) **AND** TWAIT(tg);
**IF** FWAIT(V(t')g) **AND** CF(tg) **AND** FWAIT(EVERY$_{1|t''<t'}$ t''g)=0 **THEN** $\alpha_4$(t<t') : WK(tgt');
**IF** CF(tg) **AND** FWAIT(EVERY$_1$ t'g)=0 **THEN** $\alpha_5$ : TLOCK(tg);
**IF** WK(tgt') **AND** WOUND(t'S(g))=0 **THEN** $\alpha_6$ : TLOCK(tg) **AND** FWAIT(t'g);
**IF** WK(tgt') **AND** WOUND(t'S(g)) **THEN** $\alpha_7$ : TLOCK(tg) **AND** UNLOCK(t' die);
**IF** TWAIT(tg) **AND** WOUND(tS(g)) **THEN** $\alpha_8$ : RG(g) **AND** UNLOCK(t die);
**IF** TWAIT(tg) **AND** WOUND(tS(g))=0 **THEN** $\alpha_9$ : RG(g) **AND** FWAIT(tg);
**IF** TLOCK(tg) **AND** LOCK(t'g) **THEN** $\alpha_{10}$(t'<t) : TWAIT(tg);
**IF** TLOCK(tg) **AND** LOCK(t'g) **THEN** $\alpha_{10}$(t<t') : RG(g) **AND** FWAIT(tg) **AND** TW(t'S(g));
**INITIAL**
GT(EVERY t) := 1;
RG(EVERY g) :=1;
**END**
**END**

### 2.2° Wound Treatment Module

**STUDY** Wound_Treatment_Module ;
**TYPE** TRANSACTIONS = [ t ]; GRANULES = [ g ]; SITES = [ s ]; REQUESTS = [ tg ]; AGENTS = [ ts ]; EVENTS = [ t die, t term ]; CUPTREE = [ tss' ];
**PLACE** RG : GRANULES; FWAIT, DW, LOCK : REQUESTS; TW, NW, WOUND, EW : AGENTS; UNLOCK : EVENTS; PT : CUPTREE;
**TRANS** $\beta_4$, $\beta_7$, $\beta_8$ : REQUESTS; $\beta_1$, $\beta_2$, $\beta_3$, $\beta_6$ :

```
  AGENTS; $\beta_5$ : CUPTREE;
FUNCTION S ( GRANULES ) ---> SITES;
  g ---> s;
EF;
/* EVERY$_i$ (respectively EVERY$_{i|c}$) designates the
  projection onto the i$^{th}$ component (respectively
  verifying condition c) of the function EVERY */
IF TW(ts) AND WOUND(ts)=0 THEN $\beta_1$ : NW(ts)
  AND WOUND(ts);
IF TW(ts) AND WOUND(ts) THEN $\beta_2$ : ;
IF NW(ts) AND FWAIT(EVERY$_{2|S(g)=s}$ tg)=0 THEN
  $\beta_3$ : EW(ts);
IF RG(g) AND NW(EVERY$_{2|S(g)=s}$ ts) AND
  FWAIT(tg) THEN $\beta_4$ : DW(tg);
IF PT(tss') AND EW(ts) THEN $\beta_5$ : TW(ts');
IF EW(ts) AND PT(EVERY$_3$ tss') THEN $\beta_6$ : ;
IF DW(tg) AND LOCK(EVERY$_{1|t'>t}$ t'g) AND
  FWAIT(EVERY$_{1|t'<t}$ t'g)=0 THEN $\beta_7$ : RG(g) AND
  FWAIT(tg);
IF DW(tg) AND FWAIT(EVERY$_{1|t'<t}$ t'g) THEN $\beta_8$ :
  RG(g) AND UNLOCK(t die);
IF DW(tg) AND LOCK(EVERY$_{1|t'<t}$ t'g) THEN $\beta_8$ :
  RG(g) AND UNLOCK(t die);
END
```

### 2.3° Passage from one Request to the Next Module

```
STUDYPassage_to_the_Next _Request _Module;
TYPE GRANULES = [ g ]; SITES = [ s ];
  REQUESTS = [ tg ]; AGENTS = [ ts ]; EVENTS = [
  t die, t term ]; CUPTREE = [ tss' ]; GRAREQ = [tgg'];
PLACE NEXT, TR : REQUESTS; WOUND :
  AGENTS; UNLOCK : EVENTS; PT : CUPTREE;
  AW, PW : GRAREQ;
TRANS $\mu_1$, $\mu_2$ : EVENTS; $\mu_3$, $\mu_4$, $\mu_5$, $\mu_6$, $\mu_7$ :
  GRAREQ;
FUNCTION FIN ( REQUESTS ) ---> EVENTS;
  tg ---> t term;
EF;
/* EVERY$_{2|c}$ designates the projection onto the
  second component, verifying condition c of the
  function EVERY */
IF NEXT(tg) THEN $\mu_1$(FIN(tg)) : UNLOCK(t term);
IF NEXT(tg) THEN $\mu_2$(FIN(tg)) : AW(tgg') AND
  TR(EVERY$_{2|S(g)=S(g')}$ tg');
IF AW(tgg') AND WOUND(tS(g))=0 THEN $\mu_3$ :
  PW(tgg');
IF AW(tgg') AND WOUND(tS(g)) AND
  WOUND(tS(g'))=0 THEN $\mu_4$ : TR(tg') AND
  WOUND(tS(g'));
IF AW(tgg') AND WOUND(tS(g)) AND
  WOUND(tS(g')) THEN $\mu_5$ : TR(tg');
IF PW(tgg') AND PT(tS(g')s") THEN $\mu_6$ : TR(tg')
  AND PT(tS(g)S(g'));
IF PW(tgg') AND PT(tS(g)S(g'))=0 THEN $\mu_7$ :
  TR(tg') AND PT(tS(g)S(g'));
END
```

### 2.4° Unlock Module

```
STUDY Unlock_Module ;
TYPE TRANSACTIONS = [ t ]; GRANULES = [ g ];
  REQUESTS = [ tg ]; AGENTS = [ ts ]; EVENTS = [
  t die, t term ]; CUPTREE = [ tss' ]; MESSAGES = [
  die, term ]; GRATRAME = [ t die g, t term g ];
PLACE RG, LG : GRANULES; LOCK, TR, FWAIT,
  NEXT : REQUESTS; WOUND, TW, NW, EW :
  AGENTS; UNLOCK : EVENTS; PT : CUPTREE;
  ACK : MESSAGES;
TRANS $\gamma_5$ : TRANSACTIONS; $\delta_2$ : GRANULES; $\delta_1$
  : REQUESTS; $\gamma_3$ : AGENTS; $\gamma_1$ : EVENTS; $\gamma_4$ :
  CUPTREE; $\gamma_2$ : GRATRAME;
VAR mes : MESSAGES;
FUNCTION INIT ( TRANSACTIONS ) ---> (
  REQUESTS ) ;
  t ---> tg;
EF;
/* EVERY$_2$ (respectively EVERY$_{2,3}$) designates the
  projection onto the second component
  (respectively second and third components) of the
  function EVERY */
IF UNLOCK(t mes) AND LOCK(EVERY$_2$ tg)=0
  THEN $\gamma_1$ : ACK(mes);
IF RG(g) AND UNLOCK(t mes) AND LOCK(tg)
  THEN $\gamma_2$ : LG(g) AND UNLOCK(t mes);
IF WOUND(ts) AND ACK(mes) THEN $\gamma_3$ :
  ACK(mes);
IF PT(tss') AND ACK(mes) THEN $\gamma_4$ : ACK(mes);
IF ACK(die) AND TW(EVERY$_2$ ts) AND
  NW(EVERY$_2$ ts) AND EW(EVERY$_2$ ts) AND
  PT(EVERY$_{2,3}$ tss') THEN $\gamma_5$ : TR(INIT(t));
IF LG(g) AND FWAIT(tg) THEN $\delta_1$ : RG(g) AND
  NEXT(tg) AND LOCK(tg);
IF LG(g) THEN $\delta_2$ : RG(g);
END
```

## IV - CONCLUSION

The CS-PN software, like the RDPS and GSPN software, is especially suited to the qualitative and quantitative analysis of systems. Unlike the RDPS and GSPN software, however, and with its C-DEOL specification language, coupled with advantages in conciseness, simplicity, and simplification, the CS-PN software is better adapted to site distribution and to protocol specification, verification, validation, and performance evaluation.

Furthermore, its specification, verification, validation, and evaluation procedures are more fully structured and modular, while the exploitation of results remains in the traditional mold.

## REFERENCES

[1] **M. AJMONE MARSAN, G. CHIOLA.**
On Petri Nets with Deterministic and Exponential transition firing times.
*7th European Workshop on Application and Theory of Petri Nets, pp. 151-165, Oxford, ENGLAND, July 1986.*

[2] **G.BALBO, M. AJMONE MARSAN, A. BOBBIO, C. CHIOLA, G. CONTE, A. CUMANI.**
On Petri Nets with Stochastic Timing.
*1st International Workshop on Timed Petri Nets, pp. 80-87, Torino, ITALY, July 1985.*

[3] **A. BOURGUET.**
A Petri Net Tool for Service Validation in Protocol.
*6th International Workshop on Protocol Specification, Testing and Verification, IFIP, pp. 8•17-8•28, Montreal, CANADA, June 1986.*

[4] **J.B. DUGAN, K.S. TRIVEDI, R.M. GEIST, V.F. NICOLA.**
Extended Stochastic Petri Nets: Application and Analysis.
*10th International Symposium on Computer Performance, pp. 507-519, Paris, FRANCE, December 1984.*

[5] **G. FLORIN, S. NATKIN.**
Les Réseaux de Petri Stochastiques.
*TSI, vol 4, n° 1, Dunod, pp.143-160, Paris, FRANCE, January-February 1985.*

[6] **G. FLORIN.**
Réseaux de Petri Stochastiques: Théorie et Techniques de Calcul.
*Thèse d'état, Université P.M. Curie, Paris VI, FRANCE, March 1985.*

[7] **J. GAUTIER.**
Analyse des Réseaux de Petri Colorés.
*Mémoire d'Ingénieur IIE/ CNAM, CIMSA/ SINTRA, Velizy, FRANCE, June 1986.*

[8] **S. HADDAD, C. GIRAULT.**
Algebraic Structure of Flows of a Regular Coloured Net.
*7th European Workshop on Application and Theory of Petri Nets, pp. 101-114, Oxford, ENGLAND, July 1986.*

[9] **P. HUBER, A.M. JENSEN, L.O. JEPSEN, K. JENSEN.**
Towards Reachability Trees for High-Level Petri Nets.
*DAIMI PB-174, Aarhus University, DENMARK, May 1985.*

[10] **K. JENSEN, P. HUBER, N.N. LARSEN, Ib.M. MARTINSEN.**
Petri Net Package User's Manual.
*DAIMI MD-46, Version 3.2, Aarhus University, DENMARK, March 1985.*

[11] **K. LAUTENBACH, A. PAGNONI.**
Invariance and Duality in Predicate/ Transition Net and in Coloured Nets.
*Arbeitspapiere der GMD, n° 132, Bonn, GERMANY, February 1985.*

[12] **J.P. LUGUERN.**
Protocole de Contrôle de Cohérence dans un SGBD Réparti.
*Sup Télécom, RI 86H003, ENST, Paris, FRANCE, June 1986.*

[13] **J.F. MEYER, A. MOVAGHAR, W.H. SANDERS.**
Stochastic Activity Networks: Structure, Behavior, and Application.
*1st International Workshop on Timed Petri Nets, pp. 106-115, Torino, ITALY, July 1985.*

[14] **M.K. MOLLOY.**
On the Integration of Delay and Throughput Measures in Distributed Processing Models.
*Ph.D. Dissertation, University of California, Los Angeles, USA, September 1981.*

[15] **S. NATKIN.**
Réseaux de Petri Stochastiques: Théorie et Applications.
*Thèse d'état, Université P.M. Curie, Paris VI, FRANCE, March 1985.*

[16] **M.K. VERNON, M.A. HOLLIDAY.**
Performance Analysis of Multiprocessor Cache Consistency Protocols using Generalized Timed Petri Nets.
*Performance Evaluation Review, Special Issue, vol 14, n° 1, pp. 9-17, May 1986.*

[17] **A. ZENIE.**
Colored Stochastic Petri Nets.
*1st International Workshop on Timed Petri Nets, pp. 262-271, Torino, ITALY, July 1985.*

[18] **A. ZENIE.**
Validation Qalitative et Quantitative d'un modèle SGBDR à l'aide de RdPSC.
*1st IMACS-IPAC Symposium on Modelling and Simulation for Control Lumped and Distributed Parameter Systems, pp. 467-472, Lille, FRANCE, June 1986.*

[19] **I.A. ZENIE.**
Les Réseaux de Petri Stochastiques Colorés: Application à l'Analyse des Systèmes Répartis en Temps Réel.
*Thèse de Doctorat, Université P.M. Curie - E.N.S.T. 87E017, Paris, FRANCE, June1987.*

[20] **A. ZENIE.**
The CS-PN Software: Application to the Validation and the Performance Evaluation of Distributed Systems.
*E.N.S.T. 87D005, Paris, FRANCE, September 1987.*

Dépôt légal : 3ème trimestre 1987
Imprimé à l'ENST - PARIS
ISSN : 0751-1337